Soft-Decision Reed-Solomon Decoding on Partial Response Channels

Michael K. Cheng*, Jorge Campello†, and Paul H. Siegel*,

*Center for Magnetic Recording Research,

University of California, San Diego, La Jolla, CA 92093

†Hitachi Global Storage Technologies, San Jose, CA 95193

{kcheng, psiegel}@ucsd.edu and *jorge.campello@hgst.com*

Abstract

We compare the performance of list-type soft-decision Reed-Solomon (RS) decoding algorithms to that of classical hard-decision RS decoding algorithm on Partial Response (PR) channels. The two soft-decision RS decoding approaches that we consider, the List-GMD and the Koetter-Vardy algorithm, are both based on Guruswami-Sudan's (polynomial) interpolation approach to polynomial-time list-decoding. The soft-decision RS decoders take as input symbol-based reliabilities which can be provided by a symbol-based BCJR algorithm. The symbol-based BCJR algorithm is an extension of the original BCJR algorithm and calculates the *a posteriori probability* (APP) of a block of l-bits. The complexity of the algorithm can be reduced when applied to a PR channel. We present a hybrid Viterbi-BCJR approach that can be used when only the reliability of the most-likely symbol is desired. The hybrid approach calculates the APP of the most reliable symbol without the need to run the complete forward and backward algorithm. We demonstrate through simulations that soft-decision RS decoding will lead to a lower probability of decoding error. We calculate the complexity of the list-decoding algorithms and quantify the performance and complexity tradeoff for each approach. Moreover, we show that using the symbol-based APPs will yield a lower word error rate (WER) than using the measures obtained by multiplying the bit reliabilities.

Index Terms - Soft-decision Reed-Solomon decoding, Partial Response channel, List-GMD, Koetter-Vardy decoding, Guruswami-Sudan list-decoding, Symbol-wise APP, Symbol-based BCJR, Hybrid Viterbi-BCJR

This work was supported by the Information Stroage Industry Consortium (INSIC) and the Center for Magnetic Recording Research (CMRR), UCSD.

I. INTRODUCTION

We consider a discrete memoryless source that is transmitted over a channel affected by inter-symbol interference (ISI) and additive white Gaussian noise (AWGN.) A specific scenario is the magnetic recording channel. To combat ISI in this channel, we typically use an equalizer to modify the channel read-back signal into a pre-determined partial response (PR) target [1, Chapter 9]. The equalized magnetic recording channel is also known as a PR channel and specified by a PR polynomial. For maximum likelihood detection, we can use the Viterbi algorithm applied to the trellis corresponding to the PR polynomial.

In a magnetic recording system, we often employ a concatenated coding scheme where the "inner code" is the PR channel and the outer code is an algebraic block code [2, Chapter 4]. Partial Response Maximum Likelihood (PRML) [1, Chapter 9] is a detection technique used to select the most-likely bit-sequence out of the inner channel and an algebraic outer code such as the Reed-Solomon (RS) code is used for the residual errors.

The classical bounded-distance RS decoder correctly decodes to a unique codeword if the number of errors t is less than half the minimum distance d. The initial concept of "list-decoding" was introduced independently by Elias and Wozencraft [3], [4]. List-decoding is a technique that, for a given received vector $\underline{v}$, efficiently generates a list of codewords within a Hamming distance τ from $\underline{v}$, where τ is greater than or equal to t. Guruswami and Sudan [5], [6] were the first to develop an approach that solves the list-decoding problem in polynomial time. Their approach to list-decoding consists of polynomial interpolation and factorization. In Section II, we present the List-GMD algorithm which combines Guruswami and Sudan's list-decoding with Forney's [7, Chapter 3] Generalized Minimum Distance (GMD) decoding into a sequential erasure list-decoding attempt at soft-decision Reed-Solomon decoding. Koetter and Vardy [8] have also developed a soft-decision algebraic decoding algorithm for RS codes based on Guruswami and Sudan's polynomial interpolation. Both the List-GMD and the Koetter-Vardy algorithm take as inputs symbol-wise reliability information (or APPs) from the inner PR channel. One way to approximate the symbol-wise APPs is to apply the original BCJR algorithm [9] to the inner channel and multiply the bit APPs that form a symbol. However, in using the bit-product approach, we may include invalid trellis paths in the calculation. In general, to get reliabilities for l-bit symbols we are interested in

$$Pr\left(u_{k-(l-1)}^{k} = \varphi \mid R_1^N\right), \quad \forall \varphi \in \{0,1\}^l, \tag{1}$$

and for all $k = il$, $i = 1, 2, \cdots$. This could be done by first calculating

$$Pr\left(u_1^N \mid R_1^N\right) \tag{2}$$

and then obtaining the l-dimensional marginals

$$Pr\left(u_{k-(l-1)}^k = \varphi \mid R_1^N\right) = \sum_{u_1^N \in \{0,1\}^N : u_{k-(l-1)}^k = \varphi} Pr\left(u_1^N \mid R_1^N\right) \tag{3}$$

for $\varphi \in \{0,1\}^l$. We could obtain the single bit marginals by

$$Pr\left(u_i = b \mid R_1^N\right) = \sum_{u_1^N \in \{0,1\}^N : u_i = b} Pr\left(u_1^N \mid R_1^N\right) \tag{4}$$

It is, however, infeasible to compute the conditional distribution $Pr\left(u_1^N \mid R_1^N\right)$ because the number of points in the probability distribution is exponential in N.

The symbol-based BCJR, much like the bit-based BCJR, uses the Markov properties of the source to get around the exponential complexity of calculating the l-dimensional marginals. Note that we cannot compute the l-dimensional marginals from the product of the single-bit marginals and therefore, we cannot multiply the bit APPs to calculate the symbol APPs.

To correctly calculate the l-dimensional marginals and, thus the symbol-wise APPs, we discuss a symbol-based BCJR algorithm in Section III. Our method uses the conventional bit-based trellis as in [9] and only modifies the manner in which the probability functions are updated when compared to the original BCJR algorithm. Hoeher [10] was the first to develop a method of calculating the *a posteriori probability* of a block of l consecutive bits. We however provide a description and derivation that is aimed at applications in the magnetic recording channel. We show through a concrete example that simplifications of the algorithm can be made for the case of binary-input ISI channels. We also extend Hoeher's work by introducing a reduced-complexity hybrid Viterbi-BCJR algorithm that calculates the reliability for the most-likely symbol. In Section IV, we calculate the complexity of the Guruswami-Sudan (G-S), Koetter-Vardy (K-V), and classical Berlekamp-Massey (B-M) decoding algorithms and compare the number of multiplications required to decode a codeword for each technique as a function of the code rate. In Section V, we provide simulation results that demonstrate the performance improvement of soft-decision RS decoding over hard-decision decoding. We also quantify the extra computation that is required to obtain the extra improvement. Moreover, we show that using the APPs calculated by the symbol-based BCJR algorithm

as input to the soft RS decoders will produce a lower WER than using the product of bit reliabilities. These results can help evaluate applications of soft-decoding RS list-decoding on practical systems.

II. Soft-decision Reed-Solomon decoding algorithms

A. Generalized Minimum Distance (GMD) decoding

Decoding using soft information improves performance. Forney's GMD decoding [7, Chapter 3] was an early approach at using soft channel information in the decision process. GMD decoding takes as inputs the quantized received vector $\underline{v} = \{v_1, v_2, \cdots, v_n\}$ and its associated reliability vector $\underline{r} = \{r_1, r_2, \cdots, r_n\}$, where $0 \leq r_i \leq 1$ and $1 \leq i \leq n$. Each symbol v_i in the received vector $\underline{v}$ has a corresponding reliability r_i. Denote $\underline{c} = \{c_1, c_2, \cdots, c_n\}$ to be a length n codeword and define

$$f(\hat{x}, x) = \begin{cases} +1 & x = \hat{x} \\ -1 & x \neq \hat{x} \end{cases} \tag{5}$$

and

$$\underline{r} \cdot \underline{c} = \sum_{i=1}^{n} r_i f(v_i, c_i). \tag{6}$$

Forney proved in [7, Theorems 3.1]

Theorem 1: There is at most one codeword $\underline{c}$ from a code of length n and minimum distance d for which

$$\underline{r} \cdot \underline{c} > n - d \tag{7}$$

We can sort the reliabilities in order of increasing magnitude; that is,

$$r_{i_1} \leq r_{i_2} \leq \cdots \leq r_{i_s} \leq \cdots \leq r_{i_n}. \tag{8}$$

Define the indicator vector as $\mathbf{q}_s = \{q_s(r_1), q_s(r_2), \cdots, q_s(r_{\mathrm{n}})\}$, where $1 \leq s \leq n$ and

$$q_s\left(r_{i_j}\right) = \begin{cases} 0, & 1 \leq j \leq s; \\ 1, & s+1 \leq j \leq n \end{cases} \tag{9}$$

Forney proved in [7, Theorems 3.2]

Theorem 2: If $\underline{r} \cdot \underline{c} > n - d$, then for some s, $\mathbf{q}_s \cdot \underline{c} > n - d$.

GMD decoding performs a series of erasures-and-errors hard-decision decoding on $\underline{v}$ by erasing the s least reliable symbols according to $\underline{r}$ [7, Chapter 3.2]. If a codeword satisfies the GMD criterion of (7),

it will be the unique codeword that does so and be found by the algorithm.

B. List-GMD decoding

List-GMD decoding is a combination of G-S list-decoding and Forney's GMD decoding [11]. The algorithm takes as inputs a received vector $\underline{v}$ and its associated reliability vector $\underline{r}$ which can be calculated by, for example, the symbol-based BCJR or the hybrid Viterbi-BCJR algorithms to be discussed in Section III.

The **List-GMD algorithm** is as follows:

1) *For* $s = 0, \cdots, d-1$ *do*
   a) *for each zero position in* $\mathbf{q}_s$*, erase the corresponding positioned symbol in* $\underline{v}$*. Denote this vector with* s *erased symbols* $\underline{v}_s$*.*
   b) *perform erasures-and-errors* ***G-S list-decoding*** *on* $\underline{v}_s$ *thus generating a list of candidate codewords for each decoding trial.*
2) *Select the most likely codeword from the union of the lists output by the decoding trials.*

For an (n, k, d) Reed-Solomon code with s erasures, the error correction bound of G-S erasures-and-errors decoding is given by [6, Theorem 16]:

$$\tau(s) < (n-s) - \sqrt{(n-s)(k-1)}; \tag{10}$$

that is, we can correct up to $\tau(s)$ errors when we apply erasures-and-errors list-decoding to a received vector $\underline{v}$ with s symbols erased. List-GMD decoding will perform a series of list-decodings and attempt to correct up to $\tau(s)$ errors and fill in s erasures for $s \in [0, \cdots, d-1]$.

We can skip unnecessary List-GMD trials by using the relations given in an errors-erasures table. For example, Table I provides the list errors and erasures decoding combination for the $(5, 2, 4)$ extended Reed-Solomon code over $GF(5)$. The list of codewords generated by the 2-erasures case will contain all codewords produced by the 1-erasure case, because the two cases differ by an erasure which can always be filled-in correctly. To run List-GMD on a received vector, we only have to run the algorithm once for every value of $\tau(s)$. For the cases where multiple s evaluate to the same $\tau(s)$, we only have to list-decode the case with the largest $\tau(s) + s$ combination. Moreover, the errors-erasures table only depends on n, k, and s. So we would only have to generate the table once for each code before we begin List-GMD decoding.

s	τ(s)
0	2
1	1
2	1
3	0

TABLE I
ERRORS-ERASURES TABLE FOR THE $(5, 2, 4)$ REED-SOLOMON CODE

List-GMD decoding can be computationally intensive because the algorithm involves a series of list-decodings. We can derive a Forney-like bound and use this lower bound to estimate the algorithm's performance. The minimum List-GMD threshold can be shown [12] to be

$$\underline{r} \cdot \underline{c} \geq \min_{s} \left\{ \sum_{j=1}^{\tau(s)} -r_{i_j} + \sum_{j=\tau(s)+1}^{n-s} r_{i_j} - \sum_{j=n-s+1}^{n} r_{i_j} \right\} \tag{11}$$

for $s \in [0, 1, \cdots, d-1]$. Any $\tau\,(s)$-consistent codeword will have an inner product greater than or equal to (11).

C. The Koetter-Vardy algebraic soft-decision decoding

Koetter and Vardy [8], [13], [14] developed a polynomial-time soft-decision decoding algorithm based on G-S list-decoding. Koetter and Vardy's approach uses polynomial interpolation with variable multiplicities while Sudan's technique uses polynomial interpolation with fixed multiplicities. For an (n, k, d) RS code defined over $GF\,(q)$, the Koetter-Vardy (K-V) algorithm generates a size $q \times n$ multiplicity matrix $\mathbf{M} = \{m_{i,j}\}$, $i = 1, \cdots, q$ and $j = 1, \cdots, n$, from channel posterior probabilities for a maximum possible $q \cdot n$ interpolation points. The allocation of multiplicities in the $q \times n$ matrix $\mathbf{M}$ is done by a greedy algorithm [14, algorithm A]. Each entry in $\mathbf{M}$ can be a different non-negative integer. G-S list-decoding can be viewed as a special case of the K-V algorithm with a multiplicity matrix $\mathbf{M}$ that consists of one and only one nonzero entry in each column and each entry has the same value. The K-V approach allows the more reliable entries in $\mathbf{M}$ to receive higher multiplicity values and this yields the potential for improved performance.

The complexity of the K-V algorithm depends on the Cost of the multiplicity matrix, defined by Koetter and Vardy [14] as

$$C\,(\mathbf{M}) \triangleq \frac{1}{2} \sum_{i=1}^{q} \sum_{j=1}^{n} m_{i,j}\,(m_{i,j} + 1)\,. \tag{12}$$

Let $C = C(\mathbf{M})$. The computation of the interpolating polynomial $Q_{\mathbf{M}}(X, Y)$ is equivalent to solving C linear equations. A straightforward method of solving for $Q_{\mathbf{M}}(X, Y)$ is Gaussian Elimination, however, its complexity is on the order of $O(C^3)$. In Section IV-A, we discuss Koetter's fast interpolation technique that will reduce the complexity to $O(LC^2)$ where $L \ll C$. Nielsen [15], Olshevsky and Shokrollahi [16] also have proposed different reduced complexity methods of finding $Q_{\mathbf{M}}(X, Y)$ that are on the order of $O(\kappa C^2)$ where κ is a constant. The computational complexity of Koetter-Vardy's algebraic soft-decision decoding can therefore be high. Thus, we would like to use a threshold condition to estimate its performance.

Koetter and Vardy provided a threshold condition for simulation in their paper [14, Corollary 5]. Given a codeword $\underline{c}$, define $[\underline{c}]$ as a $q \times n$ matrix. Let each row index of $[\underline{c}]$ represent an element $\zeta_i \in GF(q)$; then $[\underline{c}]_{i,j} = 1$ if $c_j = \zeta_i$ and $[\underline{c}]_{i,j} = 0$ otherwise. Define the score as

$$S_{\mathbf{M}}(\underline{c}) = \langle \mathbf{M}, [\underline{c}] \rangle = \sum_{i=1}^{q} \sum_{j=1}^{n} m_{i,j} c_{i,j} \tag{13}$$

Koetter and Vardy proved that $Q_{\mathbf{M}}(X, Y)$ has a factor $Y - f(X)$, where $f(X)$ evaluates to $\underline{c}$, if

$$S_{\mathbf{M}}(\underline{c}) \geq \sqrt{2(k-1)C}. \tag{14}$$

The threshold condition of (14) is not tight; that is, codewords that do not satisfy (14) may still be in the K-V output list.

III. A MODIFIED BCJR ALGORITHM FOR NON-BINARY SYMBOLS

A. Description and derivation

Let $\varphi \in GF(2^l)$ be an information symbol. There are l bits per symbol and we can map each symbol in $GF(2^l)$ to a distinct bit pattern; that is, $\varphi \triangleq (b_{l-1}, b_{l-2}, \cdots, b_0)$, where $b_i \in GF(2)$. Let $\underline{u} \triangleq u_{k-(l-1)}^{k}$ be the information symbol mapped to the input bit sequence from time $k-(l-1)$ to time k. Fig. 1 illustrates the index labeling. The *a posteriori probability* that the information symbol $\underline{u}$ equals φ conditioned on

the length N received sequence R_1^N is:

$$\begin{aligned} Pr\left(\underline{u}=\varphi \mid R_1^N\right) &= Pr\left(u_{k-(l-1)}=b_{l-1},\cdots,u_{k-1}=b_1,u_k=b_0 \mid R_1^N\right) \\ &= \frac{p\left(u_{k-(l-1)}=b_{l-1},\cdots,u_k=b_0,R_1^N\right)}{p\left(R_1^N\right)} \quad (15) \\ &= \frac{1}{p\left(R_1^N\right)}\sum_s\sum_{s'} p\left(\underline{u}=\varphi,s_k=s,s_{k-l}=s',R_1^N\right) \quad (16) \end{aligned}$$

The equivalence in (15) is obtained using Bayes' rule and (16) is obtained using the principle of total probability. The joint pdf can be rewritten as

$$\begin{aligned} & p\left(\underline{u}=\varphi,s_k=s,s_{k-l}=s',R_1^N\right) \\ &= p\left(\underline{u}=\varphi,s_k=s,s_{k-l}=s',R_1^{k-l},R_{k-(l-1)}^k,R_{k+1}^N\right) \quad (17) \\ &= p\left(R_{k+1}^N \mid \underline{u}=\varphi,s_k=s,s_{k-l}=s',R_1^{k-l},R_{k-(l-1)}^k\right) \\ &\quad \cdot p\left(\underline{u}=\varphi,s_k=s,s_{k-l}=s',R_1^{k-l},R_{k-(l-1)}^k\right) \quad (18) \\ &= p\left(R_{k+1}^N \mid s_k=s\right) \\ &\quad \cdot p\left(\underline{u}=\varphi,s_k=s,s_{k-l}=s',R_1^{k-l},R_{k-(l-1)}^k\right) \quad (19) \\ &= p\left(R_{k+1}^N \mid s_k=s\right) \\ &\quad \cdot p\left(\underline{u}=\varphi,s_k=s,R_{k-(l-1)}^k \mid s_{k-l}=s',R_1^{k-l}\right) \\ &\quad \cdot p\left(s_{k-l}=s',R_1^{k-l}\right) \quad (20) \\ &= p\left(R_{k+1}^N \mid s_k=s\right) \\ &\quad \cdot p\left(\underline{u}=\varphi,s_k=s,R_{k-(l-1)}^k \mid s_{k-l}=s'\right) \\ &\quad \cdot p\left(s_{k-l}=s',R_1^{k-l}\right) \quad (21) \\ &= \beta_k(s)\cdot\gamma_{(k-(l-1),k)}^{\varphi}(s',s)\cdot\alpha_{k-l}(s') \quad (22) \end{aligned}$$

The term $\beta_k(s)$ is called the backward state metric, the term $\gamma_{(k-(l-1),k)}^{\varphi}(s',s)$ is called the branch transition probability, and the term $\alpha_{k-1}(s')$ is called the forward state metric [17, Chapter 2]. Going from (17) to (18) and from (19) to (20) we apply Bayes' rule to the joint probabilities. Going from (18) to (19) and from (20) to (21) we apply to the conditional probabilities the Markov property that events after time k only depend on the current state s_k and are independent of past observations.

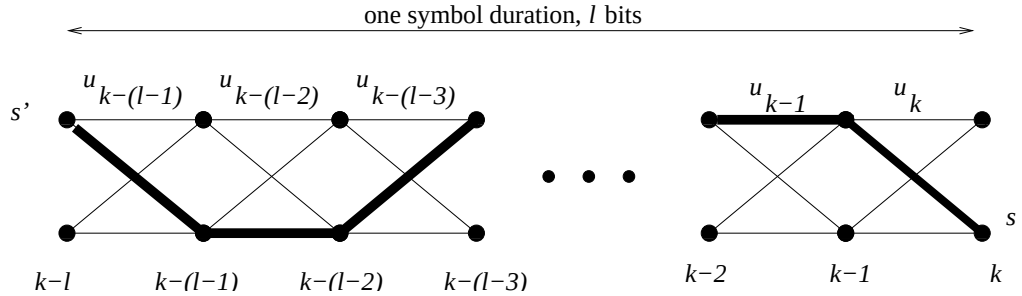

Fig. 1. A simple 2-state trellis to illustrate the indices used in the symbol-based APP derivations. The input bit sequence $\varphi \triangleq (b_{l-1}, b_{l-2}, \cdots, b_0)$ is marked by the highlighted path.

The branch transition probability can be expanded as

$$\begin{aligned}
\gamma^{\varphi}_{(k-(l-1),k)}(s', s) &= p\left(\underline{u} = \varphi, s_k = s, R^k_{k-(l-1)} \mid s_{k-l} = s'\right) && (23)\\
&= Pr\left(\underline{u} = \varphi \mid s_{k-l} = s'\right) \\
&\quad \cdot p\left(s_k = s, R^k_{k-(l-1)} \mid \underline{u} = \varphi, s_{k-l} = s'\right) && (24)\\
&= Pr\left(\underline{u} = \varphi\right) \\
&\quad \cdot Pr\left(s_k = s \mid \underline{u} = \varphi, s_{k-l} = s'\right) \\
&\quad \cdot p\left(R^k_{k-(l-1)} \mid s_k = s, \underline{u} = \varphi, s_{k-l} = s'\right) && (25)
\end{aligned}$$

From (23) to (24) we apply Bayes' rule in the following form:

$$p(x, y \mid z) = p(x \mid z) \cdot p(y \mid x, z) \tag{26}$$

where the x term is "$\underline{u} = \varphi$," the y term is "$s_k = s, R^k_{k-(l-1)}$," and the z term is "$s_{k-l} = s'$." From (24) to (25) we again apply Bayes' rule of (26) to the pdf $p\left(s_k = s, R^k_{k-(l-1)} \mid \underline{u} = \varphi, s_{k-l} = s'\right)$. Also notice that the symbol *a priori* probability is state-independent and therefore, $Pr\left(\underline{u} = \varphi \mid s_{k-l} = s'\right)$ is equivalent to $Pr\left(\underline{u} = \varphi\right)$. The second probability term in (25) is either a zero or one i.e.,

$$Pr\left(s_k = s \mid \underline{u} = \varphi, s_{k-l} = s'\right) = \begin{cases} 1, & \text{if state } s \text{ at time } k \text{ is connected} \\ & \text{to state } s' \text{ at time } k-l \\ & \text{via the input sequence } \underline{u} = \varphi; \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

Therefore,

$$\gamma^{\varphi}_{(k-(l-1),k)}\left(s',s\right) = Pr\left(\underline{u}=\varphi\right)\cdot p\left(R^{k}_{k-(l-1)} \mid s_k=s, \underline{u}=\varphi, s_{k-l}=s'\right) \tag{28}$$

if state s' at time $k-l$ and state s at time k are connected through the input sequence $\underline{u}=\varphi$. The pdf $p\left(R^{k}_{k-(l-1)} \mid s_k=s, \underline{u}=\varphi, s_{k-l}=s'\right)$ is a function of the channel characteristic; in a partial response channel with AWGN the pdf can be calculated as:

$$\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^{l} e^{\frac{-\sum_{i=0}^{l-1}\left(R_{k-i}-c_i\right)^2}{2\sigma^2}} \tag{29}$$

where σ^2 is the noise variance and $(c_{l-1}, c_{l-2}, \cdots, c_0)$ is the partial response channel output sequence that corresponds to the input sequence $\varphi \triangleq (b_{l-1}, b_{l-2}, \cdots, b_0)$.

The forward state metric $\alpha_k(s)$

$$\alpha_k(s) = \sum_{s'} \alpha_{k-1}(s')\cdot\gamma_k\left(s',s\right) \tag{30}$$

and the backward state metric $\beta_s\left(s\right)$

$$\beta_k\left(s\right) = \sum_{s'} \beta_{k+1}(s')\cdot\gamma_{k+1}(s,s') \tag{31}$$

can be updated as in the original BCJR [9].

B. Simplification for ISI channels

There are simplifications that can be made for the case of binary-input ISI channels in order to save on complexity. The simplifications come from the fact that the states represent subsequences of the input sequence. To illustrate this point, consider the so-called E2PR4 channel, with transfer function $h\left(D\right) = \left(1-D\right)\left(1+D\right)^3$. This channel has a memory of $\nu=4$ and, thus a 16-states trellis representation. We calculate the *a posteriori* probability of the all-zeros 8-bit byte ending at time k:

$$Pr\left(\underline{u}=\underline{0} \mid R_1^N\right) = \frac{1}{p\left(R_1^N\right)}\sum_{s}\sum_{s'} p\left(\underline{u}=\underline{0}, s_k=s, s_{k-8}=s', R_1^N\right) \tag{32}$$

The joint pdf can be expressed as:

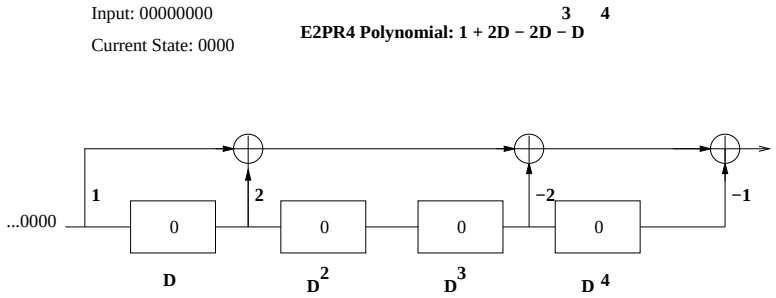

Fig. 2. E2PR4 Shift-Register Configuration

$$p\left(\underline{u}=\underline{0}, s_k=s, s_{k-8}=s', R_1^N\right)=\alpha_{k-8}\left(s'\right)\gamma_{(k-7,k)}^{00000000}\beta_k\left(s\right) \tag{33}$$

Since there is exactly one s that is reached from s' through the all-zeros symbol, the double sum in (32) becomes a single sum. Define the set of states to be $\mathcal{S}=\{S_0,S_1,\ldots,S_{2^\nu-1}\}$. For the all-zeros symbol, the ending state s at time k has to be S_0, so the sum is simply over all starting states s' at time $k-8$. The number of states per stage for the E2PR4 channel is $S=|\mathcal{S}|=16$ so there will be 16 terms in the sum. Each term takes 2 multiplications (one for α, and the other for β) plus the 8 multiplications for calculating the γ. The result is a total of

$$N_{general}=(2+8)\,S+(S-1)=175$$

operations.

For a binary input ISI channel, we notice that the state is determined by a string of length ν input bits. In Figure 2, we see that the channel state is forced back to the all-zeros state after the first $\nu=4$ bits of the all-zeros byte are shifted into the channel, regardless of the starting state m'. Therefore, we can rewrite (33) as

$$p\left(\underline{u}=\underline{0}, s_k=s, s_{k-8}=s', R_1^N\right)=\alpha_{k-4}\left(S_0\right)\gamma_{(k-3,k)}^{0000}\left(S_0,S_0\right)\beta_k\left(S_0\right) \tag{34}$$

The number of operations in (34) is 2 multiplications (one for α, and the other for β) plus the 4 multiplications for calculating the γ. The total is therefore,

$$N_{ISI}=2+4=6$$

operations. There is a factor of $175/6\approx 29$ reduction in complexity. In general, for a channel with memory

ν, we can calculate the joint pdf as:

$$\begin{aligned} p\left(\underline{u}=\varphi, s_k=s, s_{k-l}=s', R_1^N\right) & \\ &= p\left(\underline{u}=\left(b_{l-(\nu+1)},\cdots,b_0\right), s_k=s, s_{k-(l-\nu)}=s'', R_1^N\right) \\ &= \alpha_{k-(l-\nu)}\left(s''\right)\gamma_{(k-(l-(\nu+1)),k)}^{\left(b_{l-(\nu+1)},\cdots,b_0\right)}\left(s'',s\right)\beta_k\left(s\right) \end{aligned} \tag{35}$$

where s'' is the state that corresponds to the shift register configuration after an input of ν bits.

As a result, we would only need

$$N_{ISI} = 2 + (l-\nu)$$

operations, as opposed to

$$N_{general} = (2+l)\,2^{\nu} + (2^{\nu}-1)$$

operations.

C. The Hybrid Viterbi-BCJR algorithm

If we are only interested in the most-likely l-bit symbol, we can apply a variation of the symbol-based BCJR to further save on complexity. The procedure consists of *(i)* running the forward updates until the time instance $k-(l-\nu)$; *(ii)* running the backward updates until the symbol boundary at time k; *(iii)* applying the Viterbi algorithm starting at time k with the initial metric β and working our way **backwards** to time $k-(l-\nu)$; *(iv)* multiplying the Viterbi metric for each state at time $k-(l-\nu)$ by the corresponding α and choosing the state with the largest product. Since this procedure incorporates aspects of both the Viterbi algorithm and the BCJR algorithm, we call it the Hybrid Viterbi-BCJR algorithm.

The details of the algorithm are described below. It is assumed that we have a trellis for an ISI channel with memory ν; that is, there are 2^{ν} states that are in one-to-one correspondence with all 2^{ν} possible binary strings of length ν.

THE ALGORITHM

The BCJR Phase

Run the BCJR algorithm storing the forward metrics α at the time instances $k-(l-\nu)$ and the backward metrics β at time instances k, where $k=il,\ i=0,1,2,\ldots,N_s$ and N_s is the number of symbols to be detected.

The Viterbi Phase

1) **Initialization** The backwards Viterbi metrics at time k, μ_k, are initialized with the BCJR β's at time k; that is,

$$\mu_k(S_i) = \beta_i^k, \qquad i = 0, 1, \ldots, 2^\nu - 1. \tag{36}$$

2) **Propagation (The Viterbi Phase)**

For each state $s' \in \mathcal{S}$ at time $j = k-1, k-2, \ldots, k-(l-\nu)$.

a) Calculate the accumulated branch metric for the two edges connecting state s' at time j to a state at time $j+1$. Let the two states at time $j+1$ that connect to s' be s_0 and s_1 corresponding respectively to the edges with labels 0 and 1. The accumulated branch metrics are given by

$$\tilde{\mu}_j^b(s') = \gamma_{(j,j+1)}^b(s', s_b)\mu_{j+1}(s_b), \qquad b = 0, 1. \tag{37}$$

b) Compare the two accumulated branch metrics and select the largest. Let us say that the selected branch has label b^*.

c) Update the state metric and the survivor sequence q. In $q_j(s)$ we store the state at time $j+1$ corresponding to the sequence of largest metric starting at time k (with metrics given by the β's) and ending at state s at time j. The update equations are

$$\mu_j(s') = \tilde{\mu}_j^{b^*}(s') \tag{38}$$

$$q_j(s') = s_{b^*} \tag{39}$$

3) **Termination**

For each state at time $k-(l-\nu)$, calculate the overall state metrics, $\lambda(S_i)$, as

$$\lambda(S_i) = \mu_{k-(l-\nu)}(S_i)\alpha_i \tag{40}$$

Find the state s^* with the largest overall metric. The most-likely l-bit symbol, φ^*, made up of bits from positions $k-l+1$ to k is given by the ν bits corresponding to state s^* followed by the $l-\nu$ bits obtained by reading off the edge labels from the survivor sequence $q_{k-(l-\nu)}(s^*)$. The probability of the most-likely symbol is given by

$$Pr(\varphi^*|R_1^N) = \lambda(s^*)/p(R_1^N), \tag{41}$$

and $p(R_1^N)$ is obtained from the forward portion of the BCJR algorithm [9].

In the description of the algorithm, the BCJR and Viterbi phases are separated for ease of understanding. In practice, to reduce both the latency and the memory requirements, the backward BCJR propagation and the Viterbi algorithm can be performed in parallel.

IV. Complexity Analysis

In order to provide a complete tradeoff analysis of the list decoding algorithms. We calculate the complexity of Guruswami-Sudan (G-S), Koetter-Vardy (K-V), and Berlekamp-Massey (B-M) algorithms by estimating the number of multiplies required to decode an (n, k) RS codeword for each technique as a function of the code rate.

A. G-S list-decoding

The maximum error radius of G-S decoding is given by [6]

$$\tau < n - \sqrt{n\,(k-1)}.$$

The multiplicity required To decode to the maximum τ is

$$m = \left\lfloor \frac{(k-1)\,n + \sqrt{n^2\,(k-1)^2 - 4\,(\sigma^2 - n\,(k-1))}}{2\,(\sigma^2 - n\,(k-1))} \right\rfloor,$$

where $\sigma = n - \tau$. The Cost (or number of linear equations) required to solve for a codeword given the parameters is

$$C = n \left(\begin{array}{c} m+1 \\ 2 \end{array} \right).$$

The solution can be found by inverting a $C \times (C+1)$ matrix. If Gaussian Elimination is used, the complexity will be on the order of $O\,(C^3)$ or C^3 multiplies. This number can be reduced by using Koetter's faster interpolation [18].

Koetter's approach, given in Fig. 3, is a recursive procedure that constructs the interpolating polynomial $Q\,(x, y)$. The algorithm starts with a set of $L+1$ basis polynomials $G_0 = \left(1, y, y^2, \cdots, y^L\right)$ and recursively updates each polynomial in the set for C iterations. At each iteration i, the set of polynomials $G_i = (g_{i,0}, g_{i,1}, \cdots, g_{i,L})$ is partitioned into two: the elements that satisfy the next, or D^{th}_{i+1}, Hasse derivative [18], [19] evaluation constraint, J_0, and the elements that do not, J_1. If all elements satisfy the $(i+1)$th

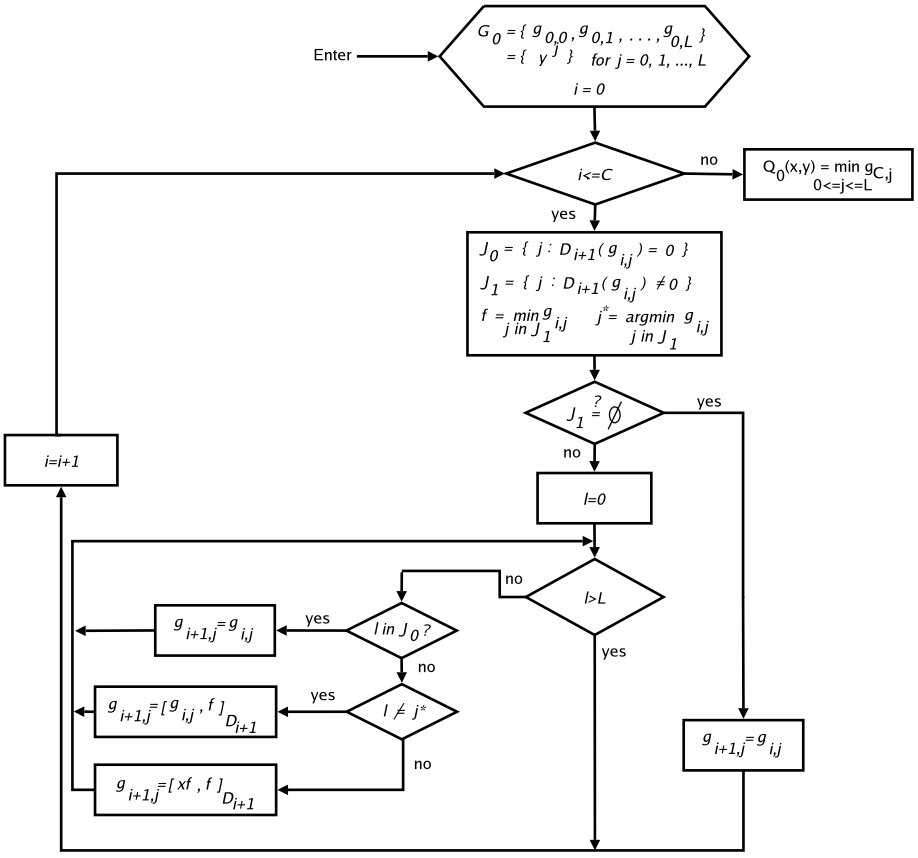

Fig. 3. The Koetter fast interpolation algorithm.

constraint, assign $G_{i+1} = G_i$ and increment i; otherwise, enter into the sub-loop to update [18] each element in the set depending on one of three conditions. The polynomial f is one with the smallest $(1, k-1)$ weighted degree [6] in J_1 and j^* is its index. McEliece showed [18] at the end of C iterations, each element in the list $G_C = (g_{C,0}, g_{C,1}, \cdots, g_{C,L})$ will satisfy all C linear constraints. We select the polynomial with the minimum degree to be $Q(x, y)$. Koetter's algorithm, therefore, has an overall complexity of $O(LC^2)$ because there are C iterations with each having $C+1$ operations to update one of $L+1$ bases. The number L is much less than C and the saving in complexity versus Gaussian Elimination is clear. Another description of Koetter's fast interpolation can be found in [20].

The Roth and Ruckenstein algorithm [21] is used to factor $Q(x, y)$ into candidate codeword polynomials. The approach has a complexity of $O\left(\left(\ell \log^2 \ell\right) k \left(n + \ell \log q\right)\right)$, where $\ell \leq L$ is the maximum y degree of $Q(x, y)$, and is much less than $O(LC^2)$. The G-S algorithm, thus, would take about $O(LC^2)$ multiplies to decode a codeword.

Algorithm 1 An approach to approximate the least Cost required to K-V decode a codeword.

1) Initialization: $i = 0$, $S = 0$, $\Delta = 0$, $\sigma = n - \tau$, $\mathbf{M} = [m_0, m_1, \cdots, m_{n-1}] = [0, 0, \cdots, 0]$
2) While $S \leq \Delta$,
   a) $m_{mod(i,n)} = m_{mod(i,n)} + 1$;
   b) $S = \sum_{l=0}^{\sigma-1} m_l$;
   c) $C = \sum_{l=0}^{n-1} \binom{m_l + 1}{2}$;
   d) $\Delta = \sqrt{2\,(k-1)\,C}$;
   e) $i = i + 1$;
3) Return S, C, and $\mathbf{M}$;

B. K-V soft-decision list-decoding

The actual Cost to K-V decode varies with many factors such as the code parameters, the channel decoder, and the the channel SNR. We avoid the need to track so many variables and therefore, estimate the K-V complexity by Algorithm 1. This approach is in fact a measure of the G-S algorithm but with weighted multiplicities.

To describe the algorithm, we first perform a hard-decision on the reliability matrix by selecting the most reliable symbol at each of the n received positions. Assume that there are τ errors in the received vector and they all occur at the last τ positions of the received word. In a round-robin fashion we allot multiplicity to each of the n positions and continue to do so until the Score of the received word is larger than $\Delta = \sqrt{2\,(k-1)\,C}$. This C can be considered as the minimum Cost required to successfully K-V decode. We again can use Koetter's fast interpolation to find $Q\,(x, y)$.

C. The Berlekamp-Massey decoding

The complexity of the Berlekamp-Massey algorithm as given in [22, Figure 7.7] is on the order of $O\,(t\,(n + 4t))$.

D. Complexity comparison

We plot the number of multiplications required to decode a codeword over $GF\,(256)$ versus the code rate for the three algorithms discussed in Fig. 4. The top figure plots the gain in error radius $\tau - t$ in using G-S list-decoding versus classical decoding. We see that the extra error correcting power of G-S list-decoding diminishes with increasing code rate. The bottom figure plots the number of multiplications required to decode a codeword. We notice that G-S decoding to the maximum τ requires many order of magnitude more computations than B-M decoding over all rates, but K-V list-decoding has a comparable

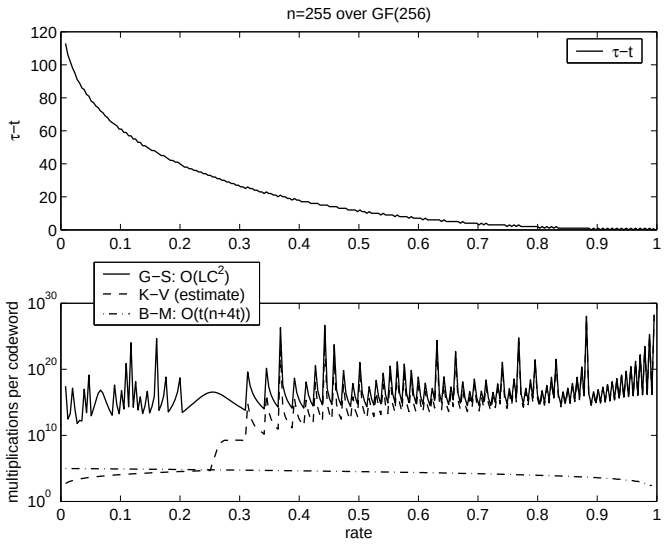

Fig. 4. Comparing the complexity of G-S, K-V, and B-M algorithms required to decode a codeword over $GF\,(256)$.

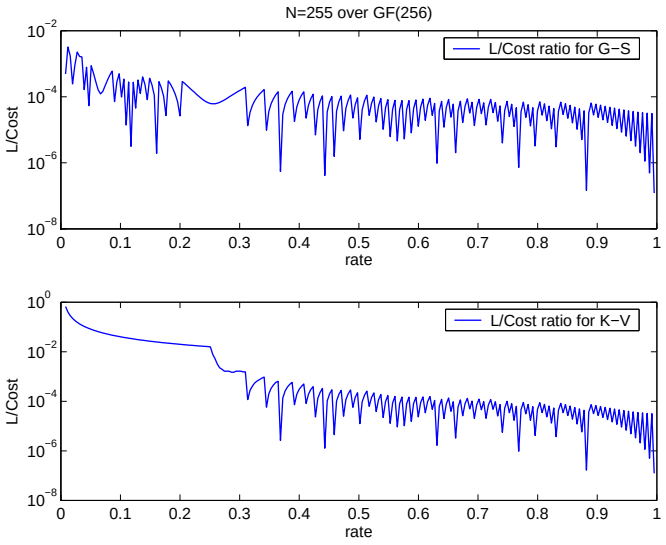

Fig. 5. L vs. C.

complexity to B-M for rates less than 1/3. This indicates that K-V decoding is an efficient alternative to B-M decoding at low rates and without a compromise in performance if the symbol reliabilities are accurate. The complexity of K-V decoding, however, converges to G-S decoding with increasing rate. The tradeoff between complexity and performance for list-decoding at high rates is small.

To verify that the number of basis polynomials L is in fact, much smaller than C, we plot their ratio in Fig. 5. This ratio also indicates the savings in using Koetter's fast interpolation over Gaussian Elimination. The reduction in complexity is more than two orders of magnitude over all rates for G-S decoding and for K-V decoding of rates greater than 1/3.

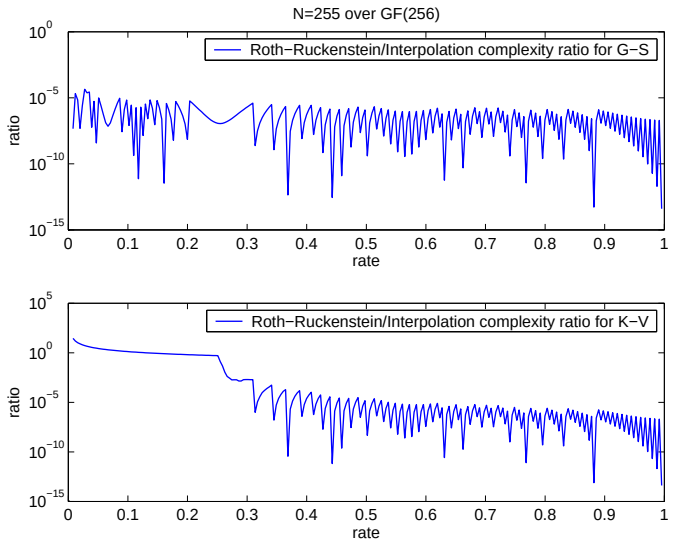

Fig. 6. Complexity of Roth-Ruckenstein over $O\left(LC^2\right)$.

To confirm that the Roth-Ruckenstein factorization has a much less complexity than bi-variate polynomial interpolation, we plot the ratio of their complexity in Fig. 6. The Roth-Ruckenstein factorization requires at least 5 orders of magnitude less computation than interpolation over all rates for G-S decoding and for K-V decoding of rates greater than 1/3.

E. Complexity control

We can reduce the complexity of the G-S algorithm by decoding to only a few errors beyond the classical bound instead of the maximum τ. Fig. 7 shows the multiplicity required to decode up to the desired error radius t. We see that the multiplicity has an exponential ramp as the error radius approaches τ, but is still reasonable for a few errors beyond t. We can then compare the complexity of decoding the RS code over $GF\left(256\right)$ versus rate for the case of $t+1$ and $t+10$ errors as given in Fig. 8. We see that for rates less than 1/3, the extra computation required to obtain an extra error correction is manageable. There is the flexibility in choosing the appropriate G-S error radius according to the computing power available in a system.

V. Simulation Results

A. Comparing the performances of Reed-Solomon decoding techniques over a partial response channel

We simulate the performances of G-S, K-V, List-GMD, GMD, and the classical bounded-distance (B-M) Reed-Solomon decoding algorithms on the EPR4 channel with transfer function $h\left(D\right)=\left(1-D\right)\left(1+D\right)^2$.

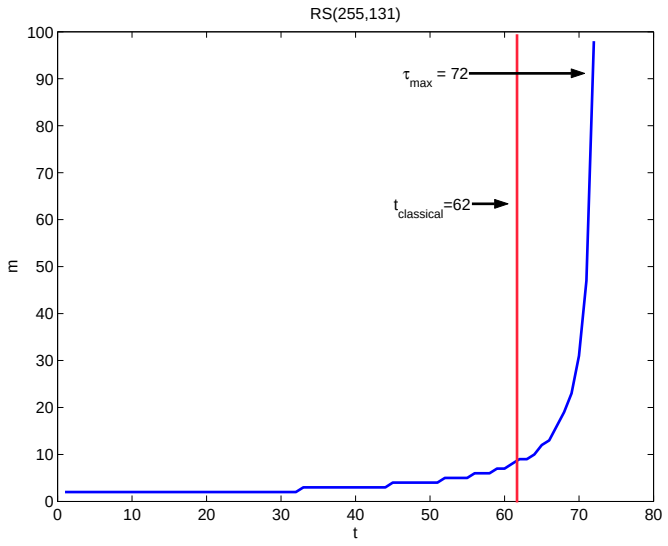

Fig. 7. The required multiplicity m to decode to an error radius t for the $RS\,(255, 131)$ code.

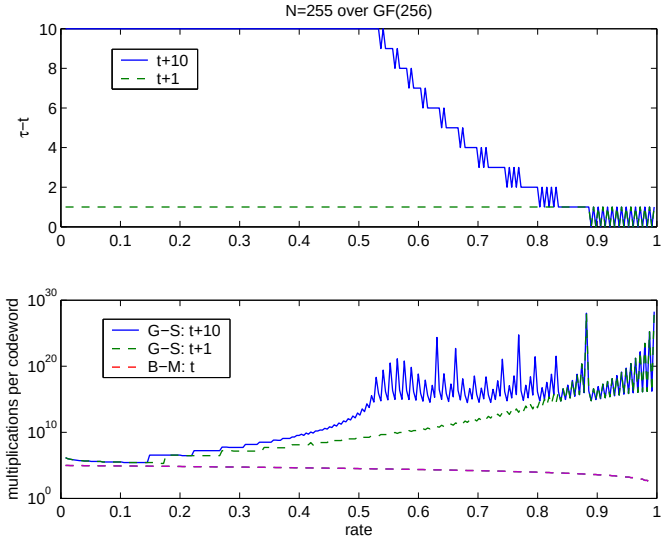

Fig. 8. Comparing the complexity of G-S decoding to only $t + 1$ and $t + 10$ versus the B-M algorithm for a RS code over $GF\,(256)$.

Our system model is shown in Fig. 9. It is a simple yet effective model that allows us to compare the various RS decoding techniques. The matrix Π^* is what Koetter and Vardy referred to as the *generalized reliability matrix* [14] and applies to channels with memory.

We first discuss the performance of G-S list-decoding. To evaluate the list-decoding performance, we can run the actual algorithm or simulate the performance of a τ-error correcting code by declaring a successful decode whenever the Hamming distance of the transmitted codeword and the received vector is less than or equal to τ. The simulation assumes that the correct codeword is always selected from the list. Nielsen [15, Chapter 3, Section 3] showed that the probability of a list with multiple codewords

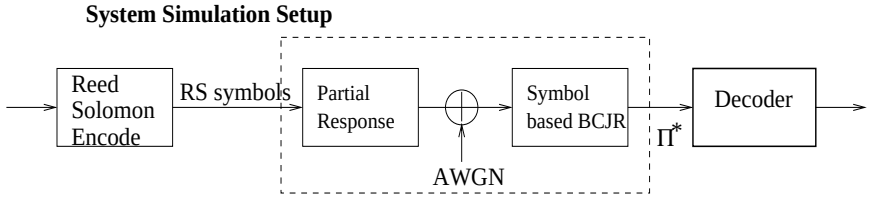

Fig. 9. Partial Response system model to compare various RS decoding techniques.

becomes smaller as the code length n and alphabet size q increase. At code parameters of practical interest in the magnetic recording setting, list-decoding will almost always produce either a one-codeword list or a zero-codeword list. For example, for a $(255, 223, 32)$ RS code $Pr\left(|\tau - \text{consistent list}| > 1\right) \leq 6.8427 \times 10^{-11}$. Furthermore, the probability of choosing incorrectly from the list of candidates is also small. For these reasons, the simulation results obtained through our assumption will closely approximate the actual decoding performance. Fig. 10(a) compares the Word Error Rate (WER) of a $(15, 7, 9)$ RS code with 4 bits per symbol over the EPR4 channel with additive white Gaussian noise (AWGN.) Fig. 10(b) compares the relative complexity of the algorithms over classical B-M decoding. The symbol reliabilities are generated by the symbol-based BCJR algorithm described in Section III. GMD provided a slight gain of 0.15 dB over classical decoding with 5X the computation while G-S list-decoding provided about 0.5 dB gain over classical decoding with about 50,000X the computation. Further performance improvement can be obtained by using soft-decision list-decoding. List-GMD provided about a 1 dB gain over GMD because we apply sequential list-decoding instead of sequential classical decoding. The performance of the K-V algorithm is affected by the total number of interpolation points used. In our simulation, we use a total of $5n$ interpolation points, where n is the length of the RS code. List-GMD provided about 1.2 dB gain over classical decoding with somewhere between 50,000X and 300,000X the computation and K-V provided the same gain with about 2000X the computation. The complexity of the K-V algorithm can be reduced by using less number of interpolation points but with a tradeoff in performance. The hybrid Viterbi-BCJR algorithm can be used in place of the symbol-based BCJR algorithm to generate the reliabilities of the most-likely symbols in the cases of GMD and List-GMD decoding.

To see that the probability of selecting an incorrect codeword is low, even for codes over small fields such as the $RS\left(7, 3\right)$ code, we plot the average list size and the incorrect selection ratio versus SNR on the EPR4 channel in Fig. 11(a). An incorrect selection is counted when the correct codeword is in the G-S generated list, but not selected as the output codeword. We see that the average list size decreases with increasing channel SNR and that the probability of selecting the wrong codeword in the list is about

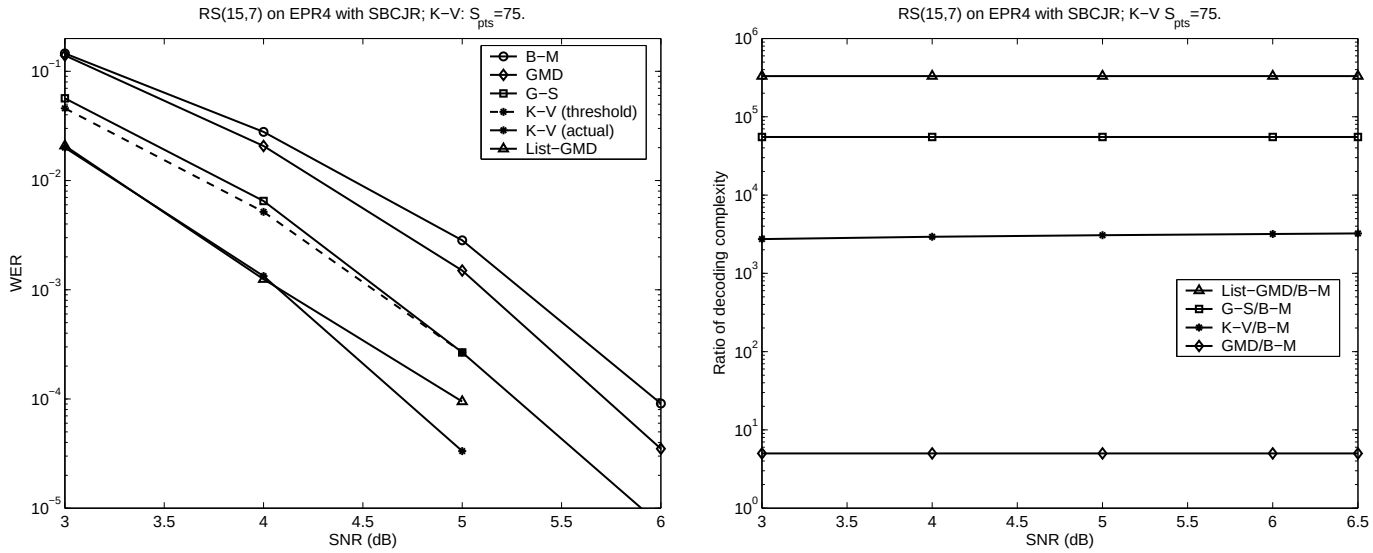

Fig. 10. (a) Performance and (b) complexity comparisons of G-S, K-V, GMD, list-GMD, and B-M decoding techniques on the EPR4 channel.

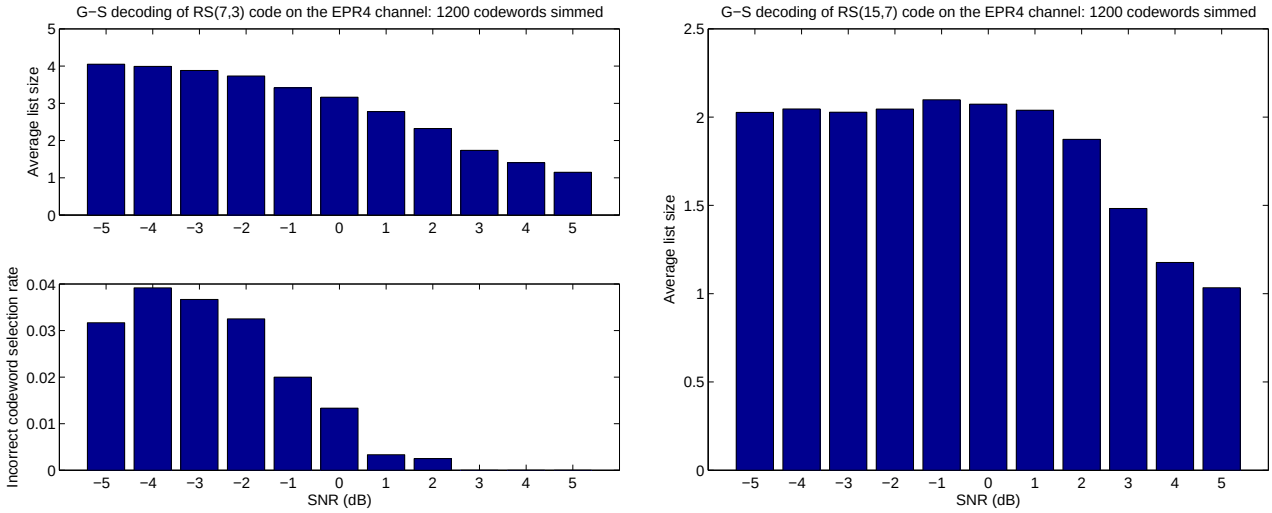

Fig. 11. Average list size and incorrect codeword selection ratio: (a) $RS\,(7,3)$ and (b) $RS\,(15,7)$ on the EPR4 channel.

4% or less. For the $RS\,(15,7)$, shown in Fig. 11(b), the average list size approaches 1 at an SNR above 5 dB and there were no incorrect codeword selections observed in decoding 1200 codewords over all SNRs. This observation agrees with Nielsen's calculation that for codes over large fields, the list size on average will be 1 and the probability of selecting an incorrect codeword from the list will be small.

In Section II-B we stated a loose lower bound for List-GMD decoding. In Fig. 12, we plot the threshold performance of K-V, List-GMD, and classical decoding for a $(255,223)$ RS code over $GF\,(256)$. The List-GMD curve is the calculated lower bound, while the K-V curve is obtained using the threshold condition (14). The actual List-GMD performance will be above the curve provided in the figure. The actual K-V

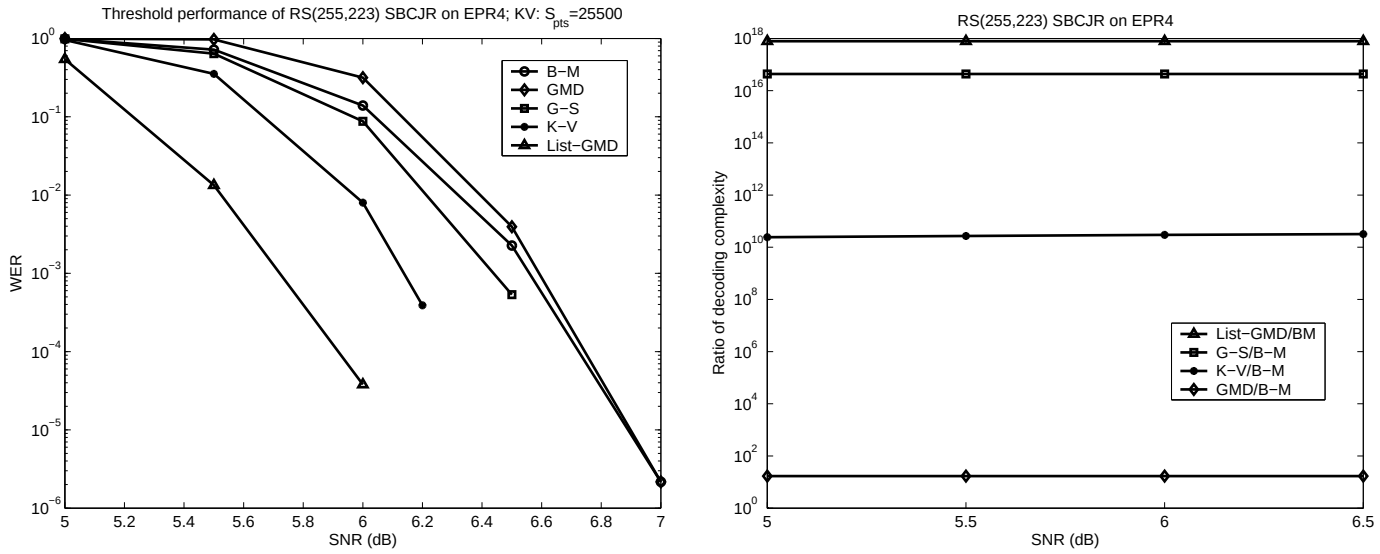

Fig. 12. (a) Threshold performance of List-GMD and Koetter-Vardy decoding of a $(255, 223)$ RS code over $GF\left(2^8\right)$. To reduce simulation time, the List-GMD performance is obtained using the lower-bound given in (11) and the K-V performance is obtained using the threshold condition given in (14). (b) A comparison of the complexity ratios over classical B-M decoding.

performance will be below the curve shown in the figure. The threshold K-V curve indicates a 0.5 dB gain and the List-GMD lower bound indicates a 0.75 dB gain over classical RS decoding. The complexity ratios over B-M decoding is given in Fig. 12(b). We see that GMD decoding has a reasonable complexity when compared to the B-M algorithm. However, G-S and K-V decoding of a codeword requires 10 orders of magnitude more computations than B-M decoding. Moreover, the flexibility in multiplicity allocation in K-V decoding allows it to have a much better performance complexity tradeoff because it can obtain a larger gain at the same complexity as G-S. Because of the high G-S complexity, List-GMD decoding is less attractive in decoding a high rate code than decoding the rate less than 1/2 code of Fig. 10. We can also vary the number of interpolation used in K-V decoding to calculate the performance and complexity tradeoff as seen in Fig. 13. Using $30n$ number of interpolation points is enough to guarantee a better performance than G-S decoding.

B. Symbol-based BCJR versus bit-product BCJR

We compare the performance of a magnetic recording system that uses the bit-product reliability measures to one that uses the symbol-wise APPs to make hard-decisions on the information symbols. The simulation platform is shown in Fig. 14 and consists of the following steps:

1) Generate a stream of random numbers according to a uniform distribution in the range of $(0, 1)$. Quantize each random number into one of 2^l equally spaced levels, where each level represents a

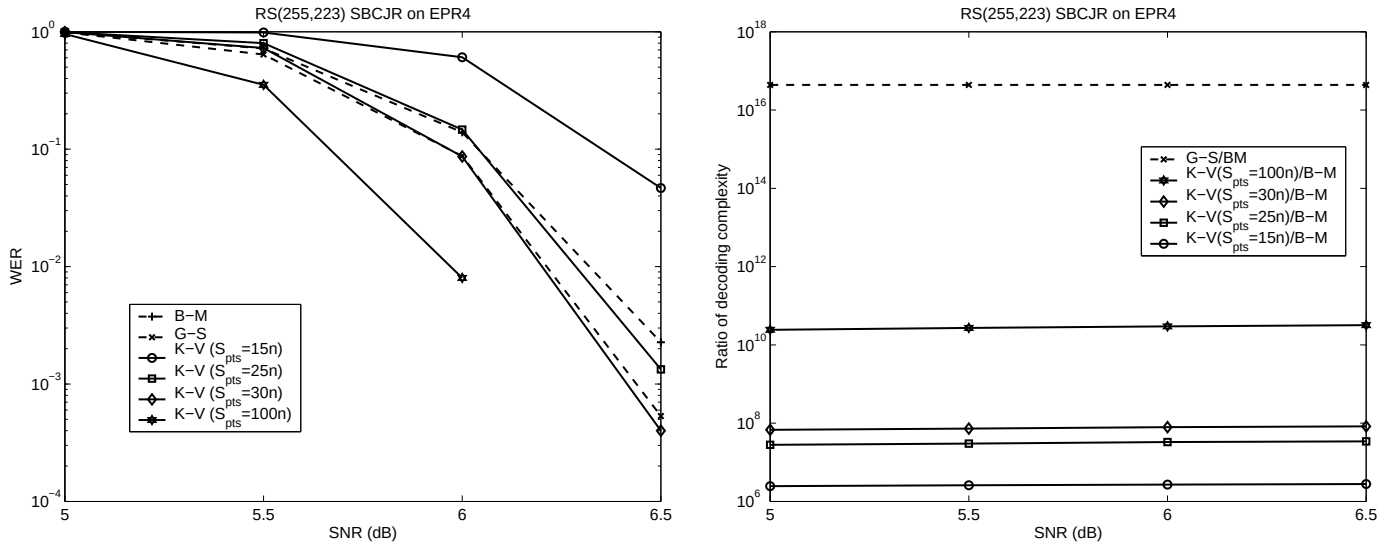

Fig. 13. (a) Threshold performance of K-V decoding according to the number of interpolation points S_{pts} used. (b) The relative complexity ratio over B-M decoding.

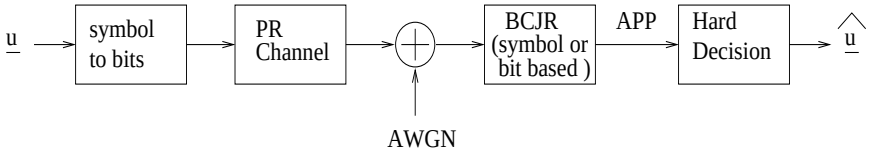

Fig. 14. Simulation setup

symbol in $GF\left(2^l\right)$, thus generating a stream of random information symbols $\underline{u}$ in $GF\left(2^l\right)$.

2) Convert the symbol stream $\underline{u}$ into bits by mapping each symbol in $GF\left(2^l\right)$ into a block of l bits.
3) Encode the bits using the polynomial specified by the PR channel.
4) Corrupt the encoded bits by adding independent and identically distributed (i.i.d.) samples of Gaussian random variables.
5) Compare the two BCJR-based approaches by
   a) applying the conventional BCJR algorithm to obtain bit APPs and take the product to generate the symbol reliability measures and make hard-decisions on the information symbols ($\underline{\widehat{u}}$) based on the reliability measures, or
   b) applying the symbol-based BCJR to generate the true symbol APPs and make hard-decisions on the information symbols ($\underline{\widehat{u}}$) based on the symbol APPs.
6) Compare the decisions ($\underline{\widehat{u}}$) with the transmitted symbols ($\underline{u}$) to calculate the symbol error rate (SER) for each approach.
7) Convert the decisions ($\underline{\widehat{u}}$) and the transmitted symbols ($\underline{u}$) into bits and calculate the bit error rate

(BER) for each approach.

We plot the simulation results generated by the EPR4 $(1-D)(1+D)^2$ channel with 8 bits per symbol in Fig. 15, the E2PR4 $(1-D)(1+D)^3$ channel with 8 bits per symbol in Fig. 16, and the EPR4 channel with 4 bits per symbol in Fig. 17. Each of these figures has two sub-figures. The sub-figure (a) plots the symbol error rate (SER) curves and the bit error rate (BER) curves generated by the symbol-based BCJR approach and the bit-product BCJR approach. The sub-figure (b) plots the reduction in SER using the symbol-based BCJR calculated by $1 - \frac{\text{SER}_{\text{SBCJR}}}{\text{SER}_{\text{bit-product}}}$. We see that in all three scenarios considered, the symbol-based BCJR produced the lowest SER and the (bit-based) BCJR produced the lowest BER.

It is interesting to note the effect that the trellis complexity and the symbol size have on the performance difference between the symbol-based and (bit-based) BCJR algorithms. If we let S be the number of states per stage and l be the number of bits per symbol, then the number of valid paths per symbol will be S independent of l and the total number of possible paths (both valid and invalid) per symbol will be S^l. We therefore would expect the SER performance improvement in using the symbol-based APPs to increase with increasing trellis complexity S and increasing bits per symbol l. The effect of trellis complexity is verified experimentally by comparing the relative SER plots in Fig. 15(b) and Fig. 16(b). Symbol-based BCJR led to a maximum of 7% reduction in SER over the bit-product BCJR on the EPR4 channel with an 8-state trellis and led to a 10% reduction in SER on the E2PR4 channel with a 16-state trellis. The effect of symbol size is confirmed by comparing the relative SER plots in Figures 15(b) and 17(b). Applying symbol-based BCJR to 4-bit symbols led to a maximum 5% reduction in SER and applying the algorithm to to 8-bit symbols led to a maximum of 7% reduction in SER. Thus, symbol-based BCJR can easily provide a better SER performance by simply treating the trellis in a different manner.

Finally, we apply the reliability information generated by the two approaches to the GMD [7] decoder and the K-V decoder. We again refer the readers to [8], [13], [14] for details. For the K-V decoder, instead of simulating the actual decoding algorithm, we used the simple threshold condition discussed in Section II-C. The simulation output generated by using a $(255, 223)$ Reed-Solomon code on the EPR4 channel is plotted in Fig. 18 for GMD decoding and K-V decoding. In GMD decoding, which uses only the reliabilities of the most-likely symbols, symbol-based BCJR led to a 0.1 dB SNR gain at a WER of 10^{-2}. In K-V decoding, which uses the reliabilities of all field symbols, symbol-based BCJR achieved a 0.4 dB SNR gain at a SER of 10^{-2}.

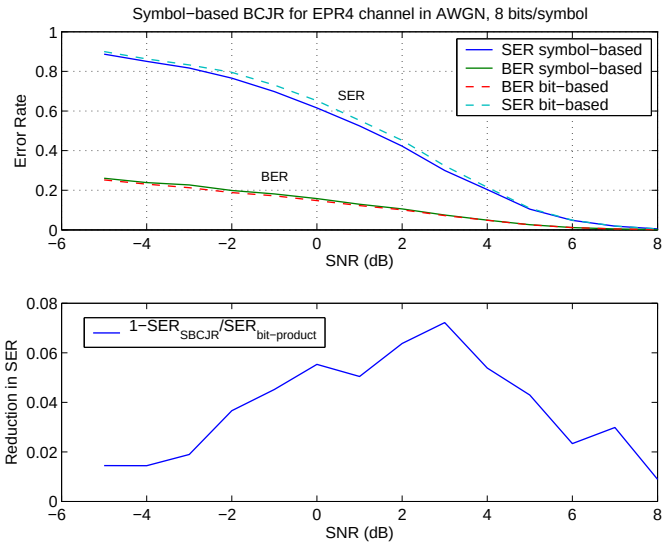

Fig. 15. (a) SER and BER for the EPR4 channel, 8 bits/symbol. *Note that to highlight the difference in the error rate curves, we have plotted the curves in linear scale.* (b) Reduction in SER using symbol-based BCJR.

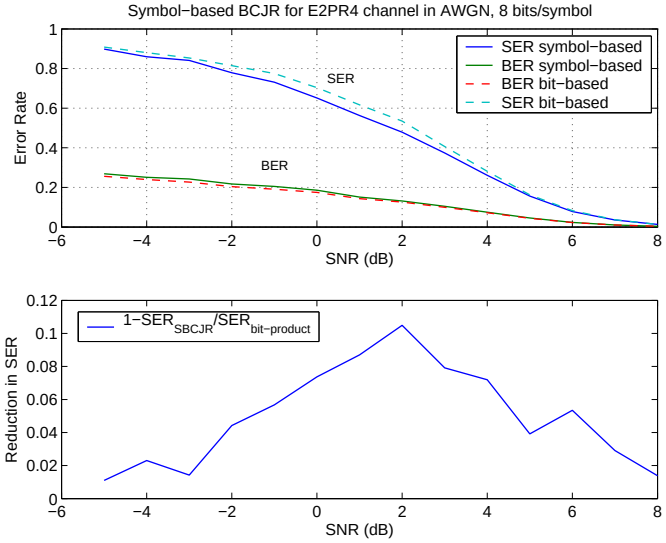

Fig. 16. (a) SER and BER for the E2PR4 channel, 8 bits/symbol. (b) Reduction in SER using symbol-based BCJR.

Symbol-based BCJR generates more accurate symbol reliabilities than taking the product of the bit-reliabilities. Applying symbol-based BCJR to soft-decision decoding of RS codes will lead to a lower probability of decoding error than applying the bit-product BCJR.

VI. Conclusions

We have discussed soft-decision Reed-Solomon decoding algorithms and their application in a magnetic recording channel. The List-GMD and the Koetter-Vardy algorithms can utilize reliability information provided by the channel. The soft-decoding algorithms outperform the classical hard-decoding algorithm

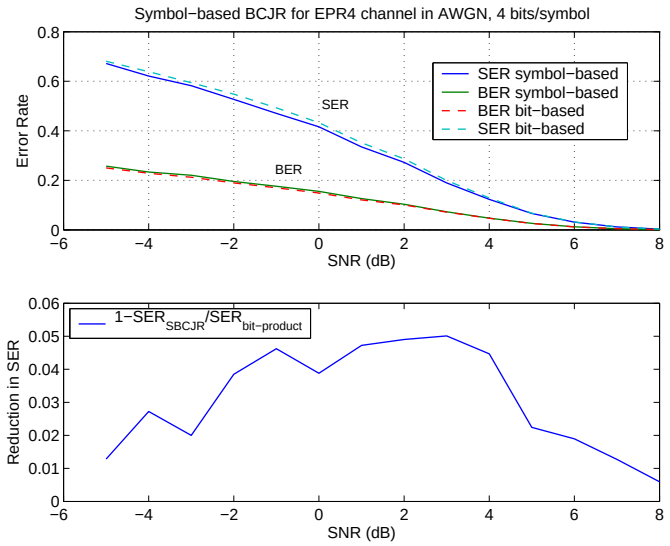

Fig. 17. (a) SER and BER for the EPR4 channel, 4 bits/symbol. (b) Reduction in SER of using symbol-based BCJR.

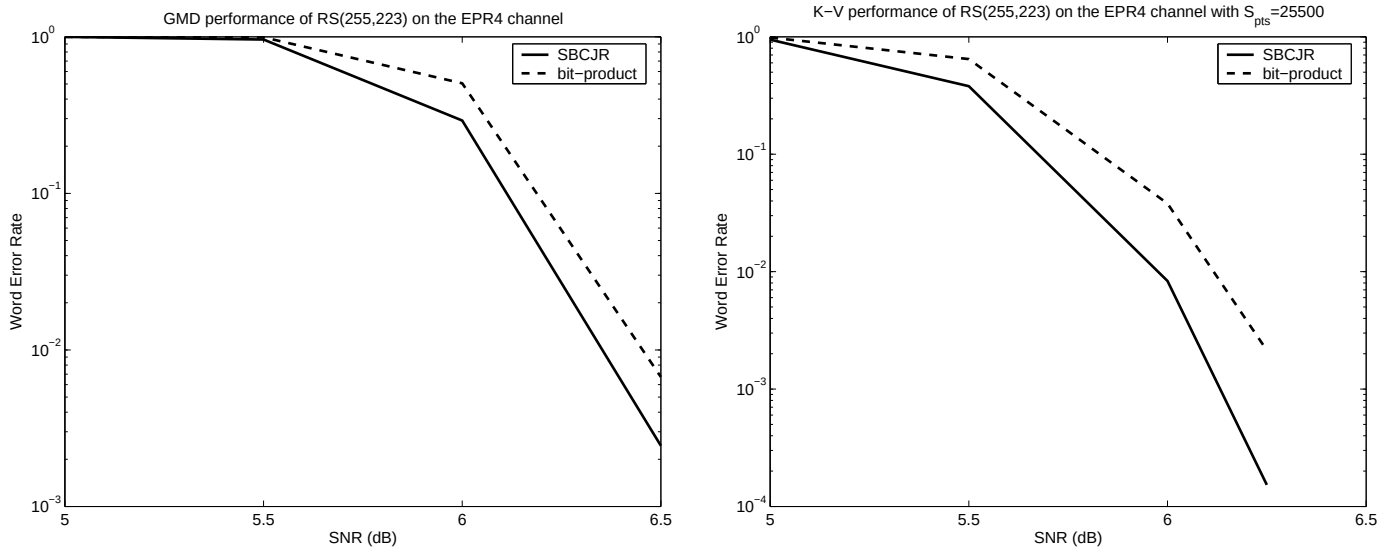

Fig. 18. Performance comparison of GMD and K-V decoding using the reliability information produced by the symbol-based BCJR algorithm versus the product of bit reliabilities.

at the cost of added complexity. We can manage the performance-complexity tradeoff in List-GMD by limiting the decoding radius τ and in Koetter-Vardy by controlling the number of interpolation points.

We presented a symbol-based BCJR algorithm that calculates the symbol-wise APPs for l-bit symbols. Our method is an extension of the original BCJR algorithm and uses the same bit-based trellis. The symbol-based BCJR technique can be used to generate symbol-wise APPs to be used by soft-decision decoding algorithms for algebraic block codes over $GF\left(2^l\right)$. Simplification of the symbol-based BCJR algorithm can be made when applying it to a binary ISI channel. We also introduced the hybrid Viterbi-BCJR algorithm which can be used when only the reliability of the most-likely symbol is desired. Simulations

over a partial response channel show that symbol decisions made by using the symbol-wise APPs yield a lower word error rate (WER) than the symbol decisions made by using the product of the bit reliabilities.

VII. Acknowledgment

Mike Cheng and Paul Siegel would like to thank Henry Pfister for discussions on the symbol-based BCJR.

REFERENCES

[1] A. M. Taratorin, *Characterization of magnetic recording systems: a practical approach*. Guzik Technical Enterprise, 1996.

[2] M. Öberg, *Turbo Coding and Decoding for Signal Transmission and Recording Systems*. PhD thesis, University of California, San Diego, La Jolla, CA, USA, April 2000.

[3] P. Elias, "List decoding for noisy channels," tech. rep., MIT, 1957.

[4] J. M. Wozencraft, "List decoding," tech. rep., MIT, 1958.

[5] M. Sudan, "Decoding of Reed-Solomon codes beyond the error correction bound," *J. Complexity*, vol. 13, pp. 180–193, Sept. 1997.

[6] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and Algebraic-Geometry codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1757–1767, Sept. 1999.

[7] G. D. Forney, Jr., *Concatenated Codes*. Cambridge, MA, USA: M.I.T. Press, 1966.

[8] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," in *Proc. IEEE Int. Symp. Information Theory*, (Sorrento, Italy), p. 61, IEEE, June 2000.

[9] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, March 1974.

[10] P. Hoeher, "Optimal subblock-by-subblock detection," *IEEE Trans. Commun.*, vol. 43, pp. 714–717, Feb. 1995.

[11] M. K. Cheng, J. Campello, and P. H. Siegel, "Soft-decision Reed-Solomon decoding on partial response channels," in *Proc. IEEE Global Telecom. Conf.*, vol. 2, (Taipei, Taiwan, ROC), pp. 1026–1030, IEEE, Nov. 2002.

[12] M. K. Cheng, *Algebraic soft-decision Reed-Solomon decoding techniques for high-density magnetic recording*. PhD thesis, University of California, San Diego, 2004.

[13] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," in *Proceedings of the 38th Annual Allerton Conference on Communication, Control, and Computing*, (Monticello, IL, USA), pp. 625–635, Oct. 2000.

[14] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 49, pp. 2809–2825, Nov. 2003.

[15] R. R. Nielsen, "Decoding AG-codes beyond half the minimum distance," Master's thesis, Danmarks Tekniske Universitet, Copenhagen, Denmark, Aug. 1998.

[16] V. Olshevsky and M. A. Shokrollahi, "A displacement approach to efficient decoding of algebraic-geometry codes," in *Proceedings of the 30th annual ACM Symposium on Theory of Computing (STOC)*, pp. 235–244, May 1999.

[17] T. V. Souvignier, *Turbo Decoding for Partial Response Channels*. PhD thesis, University of California, San Diego, 1999.

[18] R. J. McEliece, "The Guruswami-Sudan decoding algorithm for Reed-Solomon codes." to appear in the JPL publication: IPN Progress Reports; http://www.systems.caltech.edu/EE/Faculty/rjm/, April 2003.

[19] J. L. Massey and N. von Seeman, "Hasse derivatives and repeated-root cyclic codes," in *Proc. IEEE Int. Symp. Information Theory*, (Ann Arbor, USA), IEEE, 1986.

[20] W. J. Gross, F. R. Kschischang, R. Koetter, and P. G. Gulak, "A VLSI architecture for interpolation in soft-decision list-decoding of Reed-Solomon codes," in *Proceedings of the 2002 IEEE Workshop on Signal Processing Systems*, (San Diego, CA), pp. 39–44, IEEE, Oct. 2002.

[21] R. M. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *IEEE Trans. Inform. Theory*, vol. 46, pp. 246–257, Jan. 2000.

[22] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003. ISBN: 0-521-55374-1.